OpenGL: Integrated Graphics I

Session 505

OpenGL: Integrated Graphics I

Geoff Stahl
OpenGL

What You'll Learn

- Why OpenGL?
- OpenGL integration
  - Images
  - Movies

Why OpenGL?

- Fastest graphics pipeline
  - Up to 500 MB/sec via AGP memory
  - Up to 10 GB/sec GPU memory bandwidth
- Flexible API
  - Free 2D/2.5D with effects
- Hardware acceleration
  - Most advanced hardware support
- Good sample code and documentation

Why Not OpenGL?

- Accelerated 16- and 32-bit pixel depths only
  - Not optimal solution without acceleration
    - Might not be good fit for 8-bit special cases
- Not as robust an imaging model as Quartz 2D
  - PDF, fonts, lines and shapes
- Large API
  - We are here to help . . .

OpenGL Integration

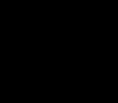
- Images
  - Render and texture anywhere
  - Image display
  - Planar image handling
- Movies
  - VR cubic texture maps
  - Movie playback

Images

- Render to anywhere
- Texture from anywhere
- Image display
- Planar images

Render to Anywhere

- Surface
- Image



Render to Surface

- On screen
  - **aglSetDrawable**
  - **NSOpenGLView**
- Off screen
  - Simply do not show window
  - Ensure proper virtual screen is set
    - Cocoa and Carbon methods

Render to Image

- Create buffer

```
pBuffer = NewPtrClear (width * height * depthBytes);
```

- Read from the back buffer

```
glReadBuffer (GL_BACK);
glReadPixels (0, 0, width, height, GL_BGRA_EXT,
                    GL_UNSIGNED_INT_8_8_8_8_REV,
                    pBuffer );
```

- Covered in depth in Game Solutions session
  - Along with saving to file

Texture From Anywhere

- File
- Image
- Surface

Texture From File

- Load image with QuickTime Image Compression API
- Build back buffer for image

```
GetGraphicsImporterForFile
                        (&fsspecImage, &giComp);
err = GraphicsImportGetNaturalBounds
                        (giComp, &rectImage);
rowStride = (rectImage.right - rectImage.left) *
                wTargetDepth / 8;
pBaseAddr = (unsigned char *) NewPtrClear (rowStride *
                        (rectImage.bottom - rectImage.top));
```

Texture From File . . .

- Build back buffer for image . . .

```
if (wTargetDepth == 32)
        QTNewGWorldFromPtr
                (&pGWImage, k32ARGBPixelFormat,
                  &rectImage, NULL, NULL, 0, pBaseAddr,
                  rowStride);
else
        QTNewGWorldFromPtr
                (&pGWImage, k16BE555PixelFormat,
                  &rectImage, NULL, NULL, 0, pBaseAddr,
                  rowStride);
if (NULL == pGWImage)
        return;
```

Texture From File . . .

- Set scaling matrix for texture

```
SetIdentityMatrix (&matrix);
ScaleMatrix (&matrix,
            X2Fix ((float) textureWidth / (float) imageWidth),
            X2Fix ((float) textureHeight / (float) imageHeight),
            X2Fix (0.0), X2Fix (0.0));
err = GraphicsImportSetMatrix(giComp, &matrix);
```

Texture From File . . .

- Scale decompress to GWorld

```
err = GraphicsImportSetGWorld
                        (giComp, pGWImage, NULL);
err = GraphicsImportSetQuality
                        (giComp, codecLosslessQuality);
hPixMap = GetGWorldPixMap (pGWImage);
if ((hPixMap) && (LockPixels (hPixMap)))
      GraphicsImportDraw (giComp);
UnlockPixels (hPixMap);
CloseComponent (giComp);
```

Texturing From Image

- Rectangle texture
- Scaling
- Slicing
- From **GWorld**
- From **NSView**

Texture Rectangle

- Check for extension

```
if (NULL != strstr ((const char *) strExtension,
"GL_EXT_texture_rectangle"))
```

- Enable texture rectangle

```
glEnable (GL_TEXTURE_RECTANGLE_EXT);
```

Texture Rectangle . . .

- Texture with correct target

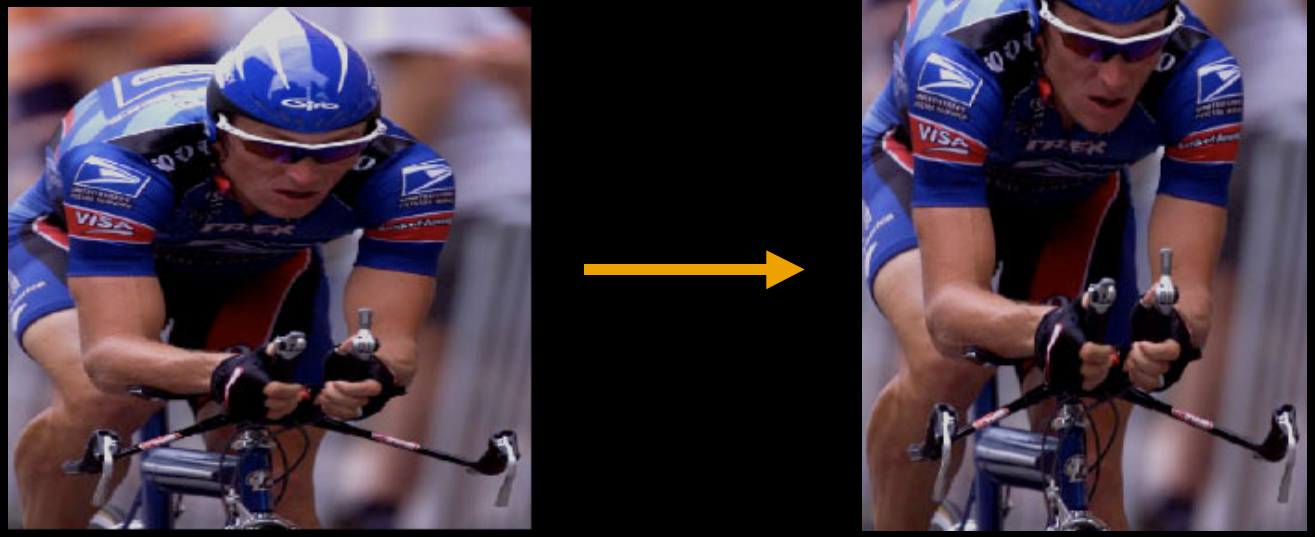
```
glTexImage2D (GL_TEXTURE_RECTANGLE_EXT,
        0, GL_RGBA, width, height, 0, GL_BGRA_EXT,
        GL_UNSIGNED_INT_8_8_8_8_REV, buffer);
```

- Use image-based coordinates vice normalized

```
glTexCoord2i (texWidth, texHeight ),
```

Scaling

- Modifies image aspect ratio on load
- Restored when drawn



Scaling . . .

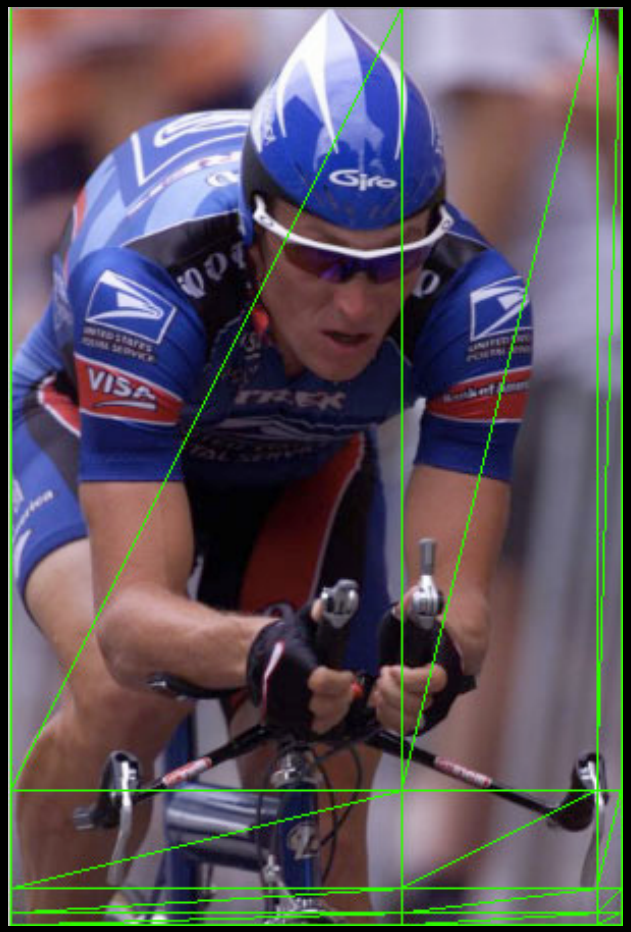
- Determine scaled texture dimensions

```
textureSize = GetTextureDimFromImageDim  (imageSize);

long GetTextureDimFromImageDim (long imageSize) {
        long i = 0, texDim = 1, imageTemp = imageDimension;
        while (imageTemp)
                { imageTemp = imageTemp >> 1; i++; }
        texDim = texDim << i;
        if (texDim > gMaxTextureSize) // if we are too big
                return gMaxTextureSize;
        else if ((texDim - imageDimension) <
                   (imageDimension - (texDim >> 1)))
                return texDim;
        else
                return (texDim >> 1);
}
```

Slicing

- Base address
  - Offset to upper left of each texture segment
- Row stride
  - Use image width
  - Set with **UNPACK_ROW_LENGTH**



Slicing . . .

- Find number of textures and generate texture objects

```
maxTextureSize = gMaxTextureSize;

textureX = GetTextureNumFromTextureDim
                                    (textureWidth, maxTextureSize);
textureY = GetTextureNumFromTextureDim
                                    (textureHeight, maxTextureSize);

pTextureName = (GLuint *) NewPtrClear (sizeof (GLuint) * textureX * textureY);
glPixelStorei (GL_UNPACK_ROW_LENGTH, textureWidth);
glGenTextures (textureX * textureY, pTextureName);
```

Slicing . . .

- Texture from each slice

```
long x, y, k = 0, offsetY = 0, offsetX = 0, currWidth, currHeight;
for (x = 0; x < textureX; x++)
{
        currWidth = GetNextTextureSize (textureWidth - offsetX,
                                                maxTextureSize);
        offsetY = 0;
        for (y = 0; y < textureY; y++)
        {
                // buffer pointer is at base + rows * row size + columns
                unsigned char * pBuffer = pImageBuffer +
                                (offsetY * textureWidth * (imageDepth >> 3)) +
                                  offsetX * (imageDepth >> 3);
```

Slicing . . .

- Texture from each slice . . .

```
        currHeight = GetNextTextureSize (textureHeight - offsetY,
                                         maxTextureSize);
        glBindTexture (GL_TEXTURE_2D, pTextureName[k++]);
        if (imageDepth == 32)
                glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA,
                              currWidth, currHeight, 0, GL_BGRA_EXT,
                              GL_UNSIGNED_INT_8_8_8_8_REV, pBuffer);
        else
                glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA,
                              currWidth, currHeight, 0, GL_BGRA_EXT,
                              GL_UNSIGNED_SHORT_1_5_5_5_REV, pBuffer);
        offsetY += currHeight;
    }
    offsetX += currWidth;
}
```

Slicing . . .

- Finding number of textures

```
long GetTextureNumFromTextureDim (long textureDim, short maxTex)
{
        long i = 0;
        if (textureDim > maxTex)
                i = textureDim / maxTex;
         textureDim -= i * maxTex;
        if (textureDim & 0x2000) i++; // 8192
        if (textureDim & 0x1000) i++; // 4096
        if (textureDim & 0x800) i++; // 2048
        if (textureDim & 0x400) i++; // 1024
        if (textureDim & 0x200) i++; // 512
        if (textureDim & 0x100) i++; // 256
        …
        return i;
}
```

Slicing . . .

- Get size of next texture from remaining image

```
long GetNextTextureSize (long textureDimension, short maxTex)
{
        if (textureDim > maxTex)
                return maxTex;
        if (textureDim & 0x2000) return 0x2000; // 8192
         if (textureDim & 0x1000) return 0x1000; // 4096
        if (textureDim & 0x800) return 0x800; // 2048
        if (textureDim & 0x400) return 0x400; // 1024
        if (textureDim & 0x200) return 0x200; // 512
        if (textureDim & 0x100) return 0x100; // 256
        …
        return 0;
}
```

Texture From GWorld

- Packed Pixels
  - **AGL_NO_RECOVERY**
  - OpenGL version 1.2 or **GL_APPLE_packed_pixel** extension

```
if (imageDepth == 32)
        glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA,
                textureWidth, textureHeight, 0,
                GL_BGRA_EXT, GL_UNSIGNED_INT_8_8_8_8_REV,
                pImageBuffer);
else
        glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA,
                textureWidth, textureHeight, 0,
                GL_BGRA_EXT, GL_UNSIGNED_SHORT_1_5_5_5_REV,
                pImageBuffer);
```

Texture From NSImage

```
NSImage                 *image
NSBitmapImageRep        *imageRep;
GLenum                  imageFormat;
unsigned char           *imageData;
long                    imageRowBytes, pixelSize;
NSSize                  imageSize;

imageRep = [ [NSBitmapImageRep alloc] initWithData:
                    [image TIFFRepresentation] ];
if ([imageRep hasAlpha] == YES) {
        if ([imageRep bitsPerPixel] != 32) return;
        imageFormat = GL_RGBA;
        pixelSize = 4;
} else {
        if ([imageRep bitsPerPixel] != 24) return;
        imageFormat = GL_RGB;
        pixelSize = 3;
}
```

Texture From NSImage . . .

```
imageRowBytes = [imageRep bytesPerRow];
imageData = [imageRep bitmapData];
imageSize = [imageRep size];

glPixelStorei (GL_UNPACK_ROW_LENGTH, sourceRowBytes/pixelSize);
glPixelStorei (GL_UNPACK_ALIGNMENT, 1);
glTexImage2D (GL_TEXTURE_2D, 0, imageFormat,
                    imageSize.width, imageSize.height, 0,
                    GL_RGBA, GL_UNSIGNED_BYTE,
                    imageData);

[sourceImageRep release];
```

Texture From Surface

- Appropriate texture target
  - **GL_TEXTURE_RECTANGLE_EXT**
  - **GL_TEXTURE_2D**
    - Use best available
- Carbon

```
aglSurfaceTexture (aglContext, textureTarget,
                   GL_RGBA8, sourceAGLContext);
```

- Cocoa

```
- createTexture:fromView:internalFormat:
```

Demo

Render to Texture

Works Across All Hardware
Texture Remains on GPU

Image Display

- Orthographic Display
  - Parallel projection
  - Ignore Z
  - Use polygon to scale
    - Could use texture coordinates
  - Use ModelView matrix to offset/rotate

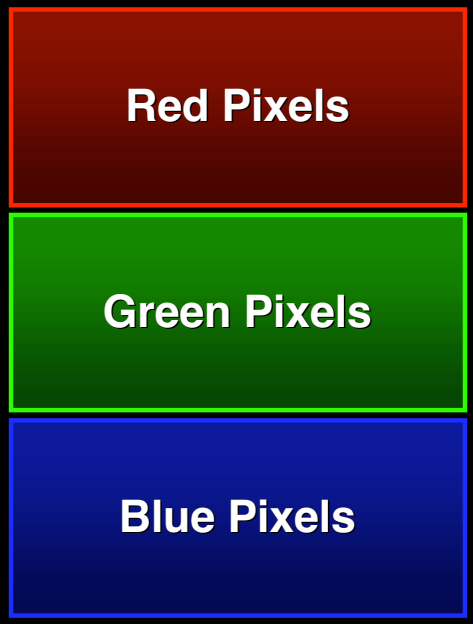
Demo

Image Texturing

Uses GPU Power 100 MB Image at 120 FPS
(12 GB/sec Effective Bandwidth)

Planar Textures

- What are they?
- Multi-pass technique



Planar Texture . . .

- Use 3 textures
  - **GL_LUMINANCE8**
  - Offset buffer pointer to start of each plane
- Bind to each texture and draw with colored polygon

```
glBindTexture (GL_TEXTURE_2D, aTextureName[0]);
glColor4f (1.0f, 0.0f, 0.0f, 1.0f); // red source color
```

- Color mask for all but the first layer drawn

```
glColorMask (1, 0, 0, 0); // red layer
```

  - Only adds component in mask
  - Other color components pass through

Demo

Planar Textures

QuickTime VR Display

- Rendered with cube map
  - Cube maps are made up of 6 square textures of the same size, representing a cube centered at the origin
- Rendering QTVR
  - Initialize and open VR movie
  - Setup VR movie and count frames
  - Read VR Frames
    - Step to each frame (F, R, Bk, L, T, Bm) and read frame
  - Render cube map

Initialize VR Movie

```
EnterMovies ();
OpenVRMovie (…);
for (long lIndex = 1; lIndex <= GetMovieTrackCount (gMovie); lIndex++) {
    Track track = GetMovieIndTrack (gMovie, lIndex);
    OSType dwType;
    GetMediaHandlerDescription (GetTrackMedia (track),  &dwType, NULL, NULL);
    if ((kQTVRQTVRType == dwType) || (kQTVRPanoramaType == dwType))
        SetTrackEnabled (track, false); // disable track
    if (VideoMediaType == dwType) {
        static Fixed width = 0, height = 0;
        Fixed newWidth = 0, newHeight = 0;
        GetTrackDimensions (track, &newWidth, &newHeight);
        if ((newWidth > width) && (newHeight > height)) {
           SetTrackEnabled (track, true);
            width =  newWidth; height = newHeight;
        }
    }
}
```

Open VR Movie

```
// setup nav stuff
anErr = NavGetFile(0, &replyNav, &optionsNav, 0, 0, 0, openList, 0);
if (anErr == noErr && replyNav.validRecord) {
    anErr = AEGetNthPtr (&(replyNav.selection), 1, typeFSS, &theKeyword,
                        &actualType, &fsspecMovie, sizeof (fsspecMovie), &actualSize);
    anErr = OpenMovieFile (&fsspecMovie, &resFile, fsRdPerm);
    if (anErr == noErr) {
        anErr = NewMovieFromFile (&gMovie, resFile, &resID, movieName,
                                  newMovieActive, &wasChanged);
        CloseMovieFile (resFile);
        if (noErr != anErr)
            DebugStr ("\pCould not get VR from file");
    } else
        DebugStr ("\pCould open VR file");
    NavDisposeReply(&replyNav);
}
```

Setup VR Movie

```
SetMoviePlayHints (gMovie, hintsHighQuality, hintsHighQuality);
GetMovieNaturalBoundsRect (gMovie, &gMovieBox);
OffsetRect (&gMovieBox,  -gMovieBox.left,  -gMovieBox.top); // move to 0, 0
SetMovieBox (gMovie, &gMovieBox);
gMovieWidth = (short) (gMovieBox.right - gMovieBox.left);
gMovieHeight = (short) (gMovieBox.bottom - gMovieBox.top);

// QTVR cube map images are all square so find scale value & offscreen size
texSize = gMovieWidth;
// set to power of two
while ((texSize >>= 1) != 1)
    shiftBits++;
do
    texSize <<= 1;
while (shiftBits--);
```

Setup VR Movie . . .

```
SetIdentityMatrix (&movieMatrix);
ScaleMatrix (&movieMatrix,
        X2Fix ((double) texSize / (double) gMovieWidth),
        X2Fix ((double) texSize / (double) gMovieHeight),
        X2Fix (0.0), X2Fix (0.0));
SetMovieMatrix (gMovie, &movieMatrix);
gMovieWidth = gMovieHeight = texSize;

// Setup off screen (as before)
gpOffscreen = …;

SetMovieGWorld (gMovie, gpOffscreen, NULL);
CountFrames ();
if (gFrameCount != 6)
    DebugStr (“\pNot Cubic QTVR Movie”);
```

Count Frames

```
OSType      whichMediaType = VideoMediaType;
short       flags = nextTimeMediaSample + nextTimeEdgeOK;
TimeValue   duration;
TimeValue   theTime = 0;

while (theTime >= 0) {
    gFrameCount++;
    GetMovieNextInterestingTime (gMovie,
                                 flags, 1, &whichMediaType,
                                 theTime, 0, &theTime, &duration);

    //  after the first interesting time, don't include the time we
    //  are currently at.
    flags = nextTimeMediaSample;
}
```

Read VR Frames

```
NextFrame ();
for (int i=0; i<6; ++i) {
   // decode QTVR order (F, R, Bk, L, T, Bm) to expected (R, L, T, Bm, Bk, F)
   switch (i) {
       case 0: traget = GL_TEXTURE_CUBE_MAP_NEGATIVE_Z_EXT; break;
       case 1: traget = GL_TEXTURE_CUBE_MAP_POSITIVE_X_EXT; break;
       case 2: traget = GL_TEXTURE_CUBE_MAP_POSITIVE_Z_EXT; break;
       case 3: traget = GL_TEXTURE_CUBE_MAP_NEGATIVE_X_EXT; break;
       case 4: traget = GL_TEXTURE_CUBE_MAP_POSITIVE_Y_EXT; break;
       case 5: traget = GL_TEXTURE_CUBE_MAP_NEGATIVE_Y_EXT; break;
    }
    glTexImage2D (target, 0, GL_RGBA,
                        gMovieWidth, gMovieHeight, 0,
                        GL_BGRA_EXT, GL_UNSIGNED_INT_8_8_8_8_REV,
                        gpBaseAddr);
    NextFrame ();
}
```

Next Frame

```
SetGWorld (gpOffscreen, NULL);
// get the next frame of the source movie
short  flags = nextTimeMediaSample;
OSType whichMediaType = VideoMediaType;
// if this is the first frame, include the frame we are currently on
if (gFrameNumber == 0)
        flags |= nextTimeEdgeOK;
// skip to the next interesting time and get the duration for that frame
GetMovieNextInterestingTime (gMovie,
        flags, 1, &whichMediaType, gMovieTime,
        0, &gMovieTime, &duration);
gFrameNumber++;
// set the time for the frame and give time to the movie toolbox
SetMovieTimeValue (gMovie,gMovieTime);
MoviesTask (gMovie, 0);
```

Draw Cube Map

```
glEnable (GL_TEXTURE_CUBE_MAP_EXT);

for (int i=0; i<6; ++i) {
    glBegin (GL_QUADS);
    for (int j=0; j<4; ++j) {
        glTexCoord2f  (texcoord[j][0], texcoord[j][1]);
        glVertex3f  (cube_vertices[cube_faces[i][j]][0],
                     cube_vertices[cube_faces[i][j]][1],
                     cube_vertices[cube_faces[i][j]][2]);
    }
    glEnd();
}

glDisable (GL_TEXTURE_CUBE_MAP_EXT);
```

Demo

QuickTime Cubic VR

QuickTime Movie Playback

- Texture Allocation
  - **QTNewGWorldFromPtr**
    - Allows QT drawing to off screen with no padding
- Texture format
  - Packed pixels for direct texture handling
- QT Synchronization
  - Minimizes texture upload
- Drawing
  - **TexSubImage2D** updates only specified texture data

QuickTime Integration

- Texture Allocation

```
rowStride = QTTextureWidth * OffScreenDepth / 8;
pBaseAddr = (unsigned char *) NewPtrClear (rowStride *
QTTextureHeight);
if (offScreenDepth == 32)
        QTNewGWorldFromPtr (&pOffscreen, k32ARGBPixelFormat,
                &rectMovie, NULL, NULL, 0, pBaseAddr , rowStride );
else
        QTNewGWorldFromPtr  (&pOffscreen, k16BE555PixelFormat,
                        &rectMovie, NULL, NULL, 0, pBaseAddr, rowStride);
SetGWorld (pOffscreen, NULL);
SetMovieGWorld (gMovie, (CGrafPtr)pOffscreen, NULL);
```

QuickTime Integration

- Texture Format (Packed pixels)

```
if (offScreenDepth == 32)
        glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA,
                        textureSize, textureSize, 0, GL_BGRA_EXT,
                      GL_UNSIGNED_INT_8_8_8_8_REV,
                        pTexture);
else
        glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA,
                        textureSize, textureSize, 0, GL_BGRA_EXT,
                        GL_UNSIGNED_SHORT_1_5_5_5_REV,
                        pTexture);
```

QuickTime Integration

- QT Synchronization

```
gMovieDrawUPP = NewMovieDrawingCompleteUPP (MovieDrawProc);
SetMovieDrawingCompleteProc (gMovie, movieDrawingCallWhenChanged,
                                        gMovieDrawUPP, &gfMovieTexUpdate);

...

pascal OSErr MovieDrawProc (Movie movie, long refcon)
{
        #pragma unused (movie)
         *((Boolean *) refcon) = true;
        return noErr;
}
```

QuickTime Integration

- Drawing

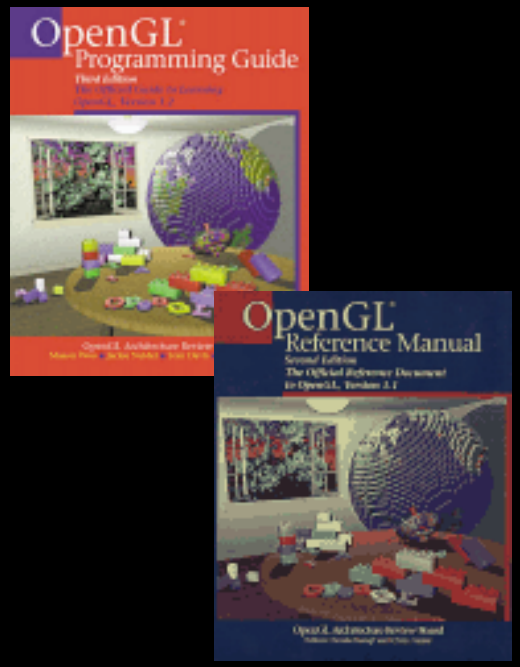
```
if (gfMovieTexUpdate) {
    if (offScreenDepth == 32)
        glTexSubImage2D (GL_TEXTURE_2D, 0, 0, 0, textureWidth, textureHeight,
                GL_BGRA_EXT, GL_UNSIGNED_INT_8_8_8_8_REV, pBaseAddr);
    else
        glTexSubImage2D (GL_TEXTURE_2D, 0, 0, 0, textureWidth, textureHeight,
                GL_BGRA_EXT, GL_UNSIGNED_SHORT_1_5_5_5_REV, pBaseAddr);
    gfMovieDraw = false;
}
MoveFrame ();
glBegin (GL_QUADS);
    glTexCoord3f (0.0, 0.0, 0.0); glColor3f (1.0, 1.0, 1.0, 1.0); glVertex3f (-1, -1, 0);
    glTexCoord3f (0.0, 1.0, 0.0); glColor3f (1.0, 1.0, 1.0, 1.0); glVertex3f (-1, 1, 0);
    glTexCoord3f (1.0, 1.0, 0.0); glColor3f (1.0, 1.0, 1.0, 1.0); glVertex3f (1, 1, 0);
    glTexCoord3f (1.0, 0.0, 0.0); glColor3f (1.0, 1.0, 1.0, 1.0); glVertex3f (1, -1, 0);
glEnd ();
```

Demo

QuickTime Integration

References

- In print
  - *OpenGL Programming Guide*
    - The red book
  - *OpenGL Reference Manual*
    - The blue book
- Online
  - **<http://www.opengl.org>**
  - **<http://lists.apple.com>**
    - Search for “opengl”
  - **<http://developer.apple.com/opengl>**



Wrap Up

- OpenGL is a powerful 2D/2.5D API
  - Provides robust and flexible tool set
  - Allows robust hardware acceleration
  - New paradigm to solve real-world problems
- Samples

  <http://developer.apple.com/samplecode/>

Roadmap

506 OpenGL: Integrated Graphics II	Room J **Wed., 10:30am**
509 ColorSync and Digital Media	Room C **Wed., 5:00pm**
511 Games Solutions: Graphics, Events, and Tidbits	Room C **Thurs., 10:30am**
512 Games Solutions: NetSprocket and OpenPlay	Room C **Thurs., 2:00pm**

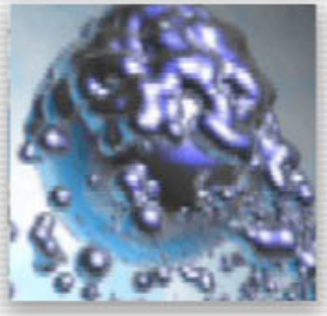
Roadmap

513 OpenGL: Advanced 3D	Room J **Thurs., 3:30pm**
514 OpenGL: Performance and Optimization	Room J **Thurs., 5:00pm**
516 Graphics and Imaging Performance Tuning	Hall 2 **Fri., 3:30pm**
FF018 Graphics and Imaging	Room J1 **Fri., 5:00pm**

Who to Contact

Sergio Mello
3D Graphics Technology Manager
sergio@apple.com

Q&A



Sergio Mello
3D Graphics Technology Manager
Worldwide Developer Relations
sergio@apple.com

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002